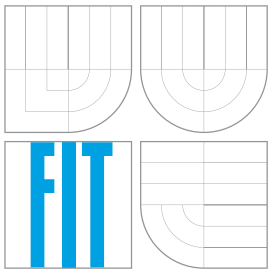
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

RASTERIZACE VEKTOROVÝCH FONTŮ NA LCD

VECTOR FONT RASTERIZATION ON LCD

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MAREK HRUŠOVSKÝ
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ GRANÁT
SUPERVISOR

BRNO 2007

Zadanie

1. Preštudujte a popíšte problematiku reprezentácie a zobrazovania vektorových fontov.
2. Preštudujte a popíšte problematiku sub-pixelového zobrazovania.
3. Vyberte a popíšte vhodný algoritmus (algoritmy) sub-pixelového zobrazovania vektorového fontu.
4. Implementujte zvolený algoritmus (algoritmy); demonštrujte správanie algoritmu za rôznych okolností.
5. Zhodnoťte dosiahnuté výsledky a navrhnite možnosti pokračovania projektu; prezentujte projekt plagátikom.

Licencia

Abstrakt

Táto bakalárska práca sa zaoberá sub-pixelovovým zobrazením vektorového písma. Všeobecne popisuje reprezentáciu a vykreslovanie vektorového písma. Stručne porovnáva výhody oproti reprezentácii bitmapového písma. Zahrňuje informácie o sub-pixelovom zobrazení písma na monitoroch typu LCD, možnostiach a problémoch, ktoré tento typ zobrazenia prináša. Obsahuje popis implementácie demonštračného programu, ktorý vykresluje písmo pomocou sub-pixelov.

Kľúčové slová

sub-pixel, Cleartype, LCD, rasterizácia, písmo.

Abstract

This bachelor's thesis deals with sub-pixel vector font representation. It generaly describes representation and rasterization of vector font and shortly compares advantages of vector font against bitmaped font. Including information about sub-pixel font representation on screen type LCD, possibilities and problems that this type of projection produces. It contains implementation description of demonstrational application, which rasterizes font with sub-pixel technology.

Keywords

sub-pixel, Cleartype, LCD, rasterization, font.

Citácia

Marek Hrušovský: Rasterizácia vektorových fontov na LCD, bakalárska práca, Brno, FIT VUT v Brne, 2007

Rasterizácia vektorových fontov na LCD

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Jiřího Granáta. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Marek Hrušovský
13. máj 2007

Poďakovanie

Autor je dlžníkom vedúcemu tejto témy, Ing. Jiřímu Granátovi, ktorý poskytol odbornú pomoc.

Obsah

Kapitola 1

Úvod

Doba, do ktorej sme sa narodili, využíva veľké množstvo informačných kanálov. Už keď sme uzreli svetlo sveta, boli sme predurčení vnímať ich tak, aby nám pomohli v našom budúcom vývoji.

V prvopočiatkoch si ľudia potrebovali vymieňať informácie. Keďže jazyk nebol dostatočne vyvinutý, musel sa vymýšľať iný spôsob ako je verbálna komunikácia. V tej dobe sa využívalo kreslenie na zem alebo na steny. Z hľadiska archivačných a transportných požiadaviek na prenos informácií nebola táto forma reprezentácie znalostí považovaná za vhodnú.

V čase egyptských pyramíd, pri brehoch rieky Níl, sa vynašiel spôsob archivovania informácií na kúsky vysušených rastlín pod názvom *papyrus*. Tento typ povrchu (média) už spĺňa základné potreby na prenos informácií, ale stále nepatrí k materiálom, ktoré dokážu vydržať večne. Od tejto doby sa *papyrus* zdokonaľoval až do jeho súčasnej podoby pod názvom *papier*.

V 15. storočí sa vývoj európskej civilizácie dostal ku Gutenbergovej kníhtlači. Takto mohla byť informácia rozmnožená bez väčšej námahy. Aj vďaka kníhtlači dnes existuje vzdelanie prístupné pre všetkých. Kníhtlač nám priniesla prvé knihy a o istý čas neskôr aj printové médiá. Doba pokročila až do 19. storočia, keď noviny začali vychádzať denne a spolu s inými technologickými pokrokmi sa stali najrýchlejším masovým spôsobom prenosu informácií.

Pre papier sa prvým vážnejším konkurentom stal audio-vizuálny prenos informácií vo forme televízie. Televízny prenos sa zahájil večernými správami a začal spájať rodiny, ktoré ich spoločne sledovali. Ale ako sa doba vyvýjala, taktiež spoločnosť mala menej času. Aby sa vyhovelo požiadavkam každého, prišla na svet archivácia. Televízny prenos sa archivoval už pri začiatkoch vysielania, ale nie každý mal ku kópii prístup. Videorekordér umožňoval vzhliadnuť želaný program aj v inom čase. Preto sa stal veľmi obľúbeným u širokej verejnosti. Nesmieme však zabúdať, že audio-vizuálna prezentácia má aj svoje nevýhody. Medzi najväčšiu patrí závislosť od elektrickej energie a nutnosť vlastniť vhodné zobrazovacie zariadenie, ktoré na svoju dobu nepatrilo k najmenším.

Moderná doba nám priniesla niečo revolučné a v súčastnosti hromadne rozšírené. Tým sa stal počítač s pripojením na internet. Je to najrýchlejší spôsob prenosu informácií s možnosťou trvácnej archivácie v celej histórii. Publikovaný internetový článok je okamžite prístupný pre každého. PDA zariadenia a mobilné telefóny sa zmestia do každého vrecka, a preto už nie je problém si v prostriedku hromadnej dopravy prečítať prednášky, alebo aktuálne spravodajstvo získané z internetu.

Aj keď obrazovkové spôsoby začali ohrozovať printové médiá, je neustále dôležité a potrebné informácie vnímať zrakom. V počítačovom svete monitorov, kde rozlíšenie nie

je dostatočne veľké v pomere ku zobrazovanej ploche a vzdialenosť očí od obrazovky nedosahuje dĺžku jedného metra, musíme čítať malý text. Tento malý text je z dôvodu uvedeného v predchádzajúcej vete zobrazený nepresne oproti jeho popisu. Keď nebudeme brať ohľad na vylepšenie zobrazeného písma pomocou zmeny jeho popisu, tak prvým spôsobom zlepšenia čitateľnosti bol anti-aliasing. Anti-aliasing síce vyhladzuje hrany a približuje zobrazenie písma k jeho definícii, avšak na druhú stranu núti oči zaostrovať na rozmazaný text. V súčastnosti začínajú na trhu prevahovať obrazovky s LCD displejom. Ich spôsob usporiadania pixelov nám dáva ďalšiu možnosť, ako text priblížit jeho originálnemu popisu a zároveň menej namáhať zrak. Tento spôsob bude v práci prezentovaný. Názov bakalárskej práce by zároveň mohol obsahovať podtitul *vylepšená čitateľnosť písma*, kvôli ktorej vznikla celá idea použitia sub-pixelového zobrazenia.

Kapitola 2

Písmo

V tejto kapitole sa čitateľ oboznámi so základnými pojmami akými sú znak a font. Ďalej bude informovaný o spôsobe zakódovania jednotlivých znakov na počítači. V tejto kapitole sa taktiež dočíta o bitmapovej a vektorovej forme uchovávania znakov. Na záver sa dozvie o štandardoch pre uchovávanie vektorového popisu znakov.

2.1 Znak

Znakom môže byť číslica, písmeno alebo aj interpunkčné znamienko. Skupinu znakov získavame z abecedy. Pre české aj slovenské písmo ďalej potrebujeme aj diakritické znamienka. Tieto sa nazývajú akcenty. Spolu s niektorými znakmi potom vytvárajú akcentovaný znak. Okrem abecedných a numerických znakov potrebujeme aj iné znaky akými sú napríklad úvodzovky. Jednotlivé písmená delíme na veľké, inak zvané aj verzálky a na malé, zvané minusky. Okrem týchto dvoch základných kategórií poznáme ešte kapitálky, ktoré majú rysy veľkého písmena, avšak ich výška je nižšia.

A b 5 ; VERZÁLKY, minusky

Obrázok 2.1: Alfanumerické, doplňujúce, verzálkové a minuskové znaky

2.2 Font

Font je skupina znakov primárne charakterizovaná svojou jednotnou typografickou podobou. Jedným z typických rysov určujúcich písmo je napr. bezpätkovosť (Arial), alebo pätkovosť (Times) písma. Zatiaľčo sa však pojem písmo vzťahuje aj na texty vznikajúce mimo digitálne (počítačové technológie), font je určený hlavne svojou počítačovou (dátovou) definíciou a reprezentáciou. Ďalšia vlastnosť písma ako rez (kurzíva, tučné atď.), či veľkosť sa nedá do pojmu font zahrnúť, pretože ide o univerzálne atribúty aplikovateľné automaticky na ktorýkoľvek typ písma. Je však možné, že písmo vyžaduje úplne špecifickú definíciu pre niektorý rez alebo druh úpravy. V tomto prípade sa samostatné definície rezu jedného druhu písma zahrňujú pod pojem font family. Názov font sa vžil taktiež pre označenie súboru obsahujúceho definíciu písma pre grafické prostredie Windows i Macintosh. Vnútorný spôsob definície sa v týchto súboroch na rôznych platformách môže líšiť. Nesmieme si však pomýliť

pojem font, ktorý je konkrétnou reprezentáciou písma(veľkosť 12pt, rez Arial atď) s pojmom font family (Typeface), ktorý je súhrnným názvom pre všetky fonty jedného rezu(rez Arial, všetky veľkosti atď.). Viac informácií nájdete na [11]

2.3 Kódovanie

Každý znak patrí do znakovej sady. Znaková sada však môže mať iba obmedzený počet znakov. V minulosti vznikla kódovacia tabuľka, ktorá zahŕňala 256 znakov, väčšinou písmená anglickej abecedy. V súčastnosti s rozšírením počítačov v celom svete bolo potrebné ku znaku reprezentovaného jedným bajtom pridať ďalší bajt. Znak je teda v počítači reprezentovaný určitou číselnou hodnotou. Táto číselná hodnota závisí od typu použitej kódovacej tabuľky.

Najpoužívanejšie kódovania:

- ASCII
- ISO-8859-x
- Windows ANSI

2.4 Bitmapový popis

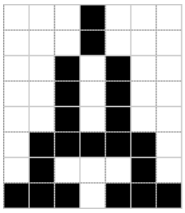
Bitmapové fonty sú jednoduchou kolekciou znakov vytvorených z rastrových obrázkov. Pre každý typeface existuje úplná zbierka obrázkov, kde každý konkrétny typ fontu obsahuje vlastnú kolekciu obrázkov. Napríklad, keď font má 3 veľkosti a môže byť písaný normálne, kurzívou a tučne, tak potom musí existovať 3×4 rôznych zbierok obrázkov (3 veľkosti, normálne, tučne, kurzívou, tučne a kurzívou). Niektoré systémy, ktoré používajú bitmapové fonty, dokážu niektoré varianty fontu odvodiť pomocou algoritmu. Napríklad počítače firmy *Apple* používajú na získanie kurzívy uhlovú deformáciu. Použitím špeciálnych algoritmov je možné taktiež dokonalejšie zmenšovanie a zväčšovanie obrázkov na získanie požadovanej veľkosti fontu.

Výhody bitmapových fontov:

- extrémne rýchle a jednoduché vykreslenie
- nezväčšené alebo nezmenšené bitmapové písmo vyzerá vždy a všade rovnako
- jednoducho sa vytvára

Nevýhody bitmapových fontov:

- vizuálna kvalita klesá so zmenšovaním, zväčšovaním, prípadne pri použití iného typu transformácie v porovnaní s vektorovým popisom
- veľký počet rôznych výstupných zariadení (s rôznymi rozlíšeniami) vyžaduje veľmi veľký počet rastrových obrázkov
- vysoká spotreba miesta na disku pre zariadenia s malou kapacitou pamäte
- k dispozícii iba malý počet písiem v niekoľkých rozlíšeniach a veľkostiach

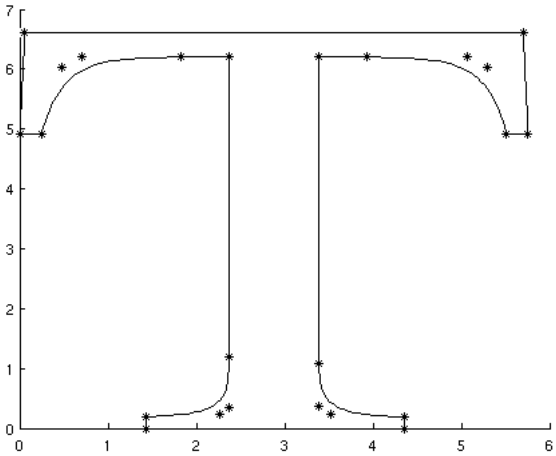


Obrázok 2.2: Raster obrázku písmena A

Prvé počítače používali výhradne bitmapové fonty. Vylepšenie hardwaru umožnilo aby boli nahradené vektorovými fontami, ktoré umožňovali použiť rôzne transformácie. Bitmapové písmo sa však stále používa dodnes vo vstavaných systémoch, kde rýchlosť a jednoduchosť patria k dôležitým vlastostiam. Bitmapové fonty sa v súčasnosti používajú v linuxovej konzole, windows recovery konzole, vo vstavaných systémoch, ale aj na starších bodových tlačiarňach. Viac sa dočítate na [10].

2.5 Vektorový popis

Vektorové fonty sú oproti bitmapovým založené na inom spôsobe uloženia. Ich tvary sú popísané bodmi, z ktorých sú vytvorené krivky a úsečky. Aby sa vektorové písmo zobrazilo na obrazovke, musí sa previesť na bitmapové. Na vytvorenie fontu je preto potreba omnoho viac znalostí ako pri vytváraní rastrov pre bitmapové písmo. Tento spôsob popisu bol prvý krát použitý v roku 1985 PostScriptovou tlačiarňou.



Obrázok 2.3: Vektorový popis písmena 'T' zložený z kriviek a úsečiek

Výhody vektorovo popísaných fontov:

- tvar popísaný krivkami a úsečkami je univerzálny, čiže pomocou matematickej transformácie každého váhového bodu sa dá získať písmo rôznych veľkostí
- výrazné šetrenie pamäte

Nevýhody vektorovo popísaných fontov:

- Poznáme obrysy písma, ale nemáme oblasť vyplnenú. Z tohto dôvodu sa vykonáva rasterizácia, ktorá nám vytvorí bitmapový font v požadovanom rozlíšení a v požadovanej veľkosti.
- Problém s vyplňovaním (prevodom obrysov na sadu zaplnených malých plôch) nastáva u zariadení s malým rozlíšením, kde je problém rozhodnúť, či určitý pixel vyfarbiť alebo nie.
- Rasterizácia (prevod vektorového popisu do rastrového obrázka) v niektorých prípadoch zlyháva. Napríklad pri rasterizácii písmena 'm', kde sa môže stať, že nožičky majú rôznu šírku. Takýto jav nastáva pri vykreslovaní malého písma a pôsobí veľmi rušivo pri čítaní.
- Pri niektorých fontoch môže veľká zmena veľkosti pôsobiť neprirodzene oproti popisu, pre ktorý bol tento font navrhnutý.
- Je požadovaný väčší výpočetný výkon z dôvodu rasterizácie.

Métody pre zkvalitnenie:

1. V popise fontu sú zahrnuté dodatočné informácie pre renderovanie (rasterizáciu). Napríklad pre malé veľkosti písma môžu byť v popise priložené rastrové obrázky.
2. *hinting* – zarovnávanie textu, ktorý sa ide rasterizovať, s rastrovou mriežkou.
3. *smart outlines* – pri zmenách veľkosti sa mení taktiež tvar obrysu a tým pádom aj celý vektorový popis.
4. *anti-aliasing* – pomocou tejto metódy dochádza k *vyhladzovaniu* zubatých okrajov.

Viac informácií nájdete na [12].

2.6 Vektorové formáty popisu znakov

Formát popisu znakov definuje aký typ kriviek má byť použitý pre popis fontov. Dve staršie formy popisu kombinujú svoje výhody v novšom type popisu OpenType. Aj keď sa môže zdať, že najstarší PostScript patrí do „starého železa", stále zaujíma významnú pozíciu na poli DTP (Desktop Publishing) systémov, ktoré sa používajú na tvorbu dennej tlače.

2.6.1 PostScript

PostScript je programovací jazyk, ktorý vyvinula spoločnosť Parc, výzkumná časť firmy Xerox pod vedením Johna Warnocka, ktorý sa neskôr stal spoluzakladateľom firmy Adobe. PostScript Type 1 je špeciálna forma programu, ktorá sa používa na popis písiem. Keď sa teda povie *postscriptové písmo*, myslí sa tým písmo vo formáte PostScript Type 1, ktorý je štandardom pre digitálne písma ISO 9541. Písmo vo formáte Type 1 je následne rasterizované buď PostScriptovým interpretom, alebo pomocou Adobe Type Manageru. ATM sa používa pre výstup na obrazovku a má v sebe podporu pre všetky hromadne používané operačné systémy (Linux, Mac OS, Windows, Unix). V súčasnosti operačné systémy natívne podporujú Type 1 fonty.

Sila PostScriptu spočíva v jeho práci s grafikou. Grafiky sa týka zhruba 30 percent príkazov. S textovými objektmi sa pracuje rovnako ako s inými grafickými prvkami. Na popis fontov sa teda používajú úsečky a Beziérove kubiky. Postscriptové písma Type 1 navyše ako prvé začali používať horizontálny a vertikálny hinting na vylepšenie rasterizácie.

Najväčšou nevýhodou písiem vo formáte PostScript je, že štandardne je adresovateľných len 255 znakov a z nich prvých 32 je obsadených podľa normy. Písmo v tomto formáte môže obsahovať aj viac ako 255 znakov, ale nastáva problém, že niektoré aplikácie nebudú vedieť ako s týmto množstvom znakov pracovať. Problematickou časťou je tzv. *encoding vector*, ktorý mapuje jednotlivé znaky na pozície v rozmedzí 32-255. Väčšina aplikácií komunikuje prostredníctvom *encoding vectoru* a vidia len týchto 255 znakov. Existujú však aplikácie, napríklad Adobe InDesign, ktoré túto informáciu nepoužívajú a sú schopné korektne pracovať s každým znakom v písme. Postscriptové písma pod Windows sa skladajú z niekoľkých súborov. Keď písmo obsahuje tri subory, tak sú to:

1. *.AFM, ktorý obsahuje metriku (šírka znakov) a kerning – informácie podľa ktorých sa riadi vzdialenosť jednotlivých znakov od seba.
2. *.INF, ktorý obsahuje dáta potrebné k inštalácii.
3. *.PFM, ktorý sa vytvára v priebehu inštalácie a obsahuje dáta z predchádzajúcich súborov.

Postscriptové písma na Mac OS sa skladajú z dvoch častí, tzv. *kufríka* (Suitcase), ktorý obsahuje skoro zhodné dáta ako súbor *.PFM a *tlačiarne*, teda súboru, ktorý obsahuje krivky. Viac informácií na [15] a [9].

2.6.2 TrueType

TrueType je formát písma, vytvorený firmou Apple Computer v roku 1991. Tento formát bol odpoveďou na patent Type 1 zo strany Adobe. Technológia písiem TrueType sa skladá z dvoch častí. Písma samotného a rasterizátoru, čo je program, ktorý je zabudovaný priamo do operačného systému. Obidve komponenty sú nutné k tomu, aby bolo písmo rovnako zobrazené ako na monitore, tak aj na tlačiarni.

Časti na ktorých závisí ako vyzerajú jednotlivé znaky:

- program, ktorý používa písmo
- písmo samotné
- rasterizér

Prax ukázala, že rasterizér má vážne nedostatky pri generovaní znakov v malých veľkostiach a rada ďalších problémov znemožnila rozšírenie TrueType písiem do profesionálneho Desktop Publishing prostredia, a preto sa používajú väčšinou v kancelárskych softwarových balíkoch.

TrueType písma využívajú dokonalejší hinting, ktorý zachováva správny vzhľad znakov pri menších veľkostiach písma. Hinting sa používa najmä na zariadeniach s nízkym rozlíšením ako sú monitory, ale pri tlačiarňach nemá žiadny význam. Pomocou hintingu sa dá dosiahnuť na obrazovke rovnaká kvalita zobrazenia akú majú bitmapové písma, avšak množstvo času stráveného vývojármi na korekciu písma robí túto technológiu zaujímavú len pre veľké korporácie, ktoré sú schopné za tento čas zaplatiť. Na trhu sa preto nachádza množstvo fontov, ktorých kvalita je diskutabilná.

Na rozdiel od postscriptových fontov používajú na svoj popis kvadratické B-splajny. Konverzia medzi Beziérovu kubikou a kvadratickým B-splajnom je nedokonalá. Aj keď B-splajny sú podmnožinou Beziérových kriviek, dochádza k malým zaokrúhlovacím chybám nezávisle na smere konverzie. Avšak chyby sú väčšie v prípade prevodu z PostScriptu do TrueType formátu ako naopak. Dôležitejšie však je, že sa neprenášajú hintingové informácie. Viac informácií nájdete na [14] a [9].

2.6.3 OpenType

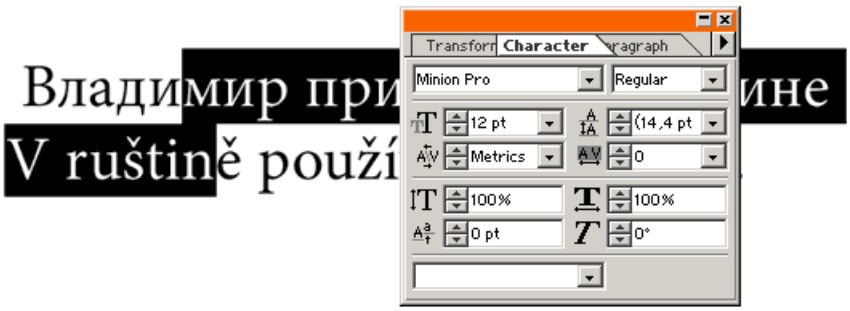
OpenType je nový formát písma, ktorý bol vytvorený spoločnosťami Adobe a Microsoft. Prvá špecifikácia bola vydaná už v roku 1997, avšak prvé OpenType písma boli dodané na trh až v roku 2000. Písma vo formáte OpenType riešia základné problémy použitia písiem vo viacjazyčnom a multi-platformovom použití.

Dôvodov vzniku bolo hneď niekoľko. Dve základné technologie písiem TrueType a Type 1 neboli schopné uspokojiť nároky používateľov. Ani jedna z technológií nie je nezávislá na operačnom systéme (z pohľadu užívateľa) a pre každý OS tak musí existovať špecifický variant fontu. Vývojovo mladší formát TrueType trpí nízkou kompatibilitou s jazykom PostScript a formátom PDF. Postscriptové písma navyše neumožňujú efektívne pracovať s viac než 256 znakmi, takže pre každý jazykový región musí existovať špeciálna verzia fontu.

Výhody

- Rovnaký OpenType font funguje bez zmeny pod OS Windows aj Mac OS.
- Podpora medzinárodných znakových sád. Vďaka kódovaniu Unicode môže byť v jednom fonte viac než 65 000 znakov. V našich končinách tak odpadá nutnosť používať *západnú* a *východnú* znakovú sadu.
- Dokonalejšia ochrana proti nelegálnemu používaniu na Internete. Tvorca písma označí svoj font ,,copyrightom" a digitálnym podpisom. Font musí byť taktiež digitálne podpísaný neskorším užívateľom, ktorý začlenil daný font do ,,layoutu" vytvorených stránok. Digitálny podpis umožňuje overiť autora písma a overí oprávnenosť užívateľa toto písmo používať. Verifikáciu digitálneho podpisu vykonáva ,,browser", ktorý podľa výsledku preverenia digitálneho podpisu rozhodne o zobrazení daného písma. Fonty sú chránené aj proti nedovoleným zásahom do dizajnu písma. Ochrana sa netýka voľne šíriteľných písiem.
- Vyspelé typografické schopnosti. Aj napriek rozvoju digitálneho publikovania a predtlačiarňovej prípravy, počítačová typografia stále nedosahovala kvalitu a možnosti klasickej sadzby. Keď grafik túžil po typograficky dokonalom vzhľade publikácie, bol nútený upravovať niektoré atribúty ručne, pre každé písmeno. OpenType je preto navrhnutý tak, aby grafikovi poskytoval pri práci maximálne pohodlie. V OpenType sú nadefinované kontextové situácie, za ktorých font uskutoční na prianie užívateľa náhradu jedného znaku za iný.

Súborový formát OpenType je rozšírením špecifikácie TrueType formátu. OpenType font je teda definovaný v jednom kompaktnom súbore štrukturovanom do systému tabuliek, v ktorých sú uložené všetky definície obrysov, kódovania, metrík atď. V niektorej literatúre sa pre OpenType používa označenie TrueTypeOpen 2.0. V skutočnosti existujú dve verzie OpenType formátu.



Obrázok 2.4: Ukážka zjednodušenej možnosti editácie atribútov pre viac znakov. Autor: www.printing.cz

1. Verzia s postscriptovými obrysmi znakov a príponou *.otf.
2. Verzia s TrueType obrysmi znakov a príponou *.ttf.

Pre verziu *.otf však musí byť v operačnom systéme aj rasterizér postscriptových písiem.

Obrázok 2.5: Automatická náhrada znakov za iné znaky. Čiernou farbou sú vyznačené nahradené znaky a červenou farbou sú vyznačené pôvodné znaky. Autor: www.printing.cz

Viac informácií nájdete na [8] a [13].

Kapitola 3

Rasterizácia

Vektorové písmo je v anglickej terminológií označované ako obrysové písmo. Obrysy sú v súčasnosti štandardom pre uchovávanie vzhľadu fontu. V nasledujúcej kapitole sa čitateľ oboznámi o spôsobe zápisu kriviek. Ďalej sa dočíta kroky nutné v procese prevodu obrysov na obrazovku.

3.1 Obrysový popis

Medzi prvé obrysovo popisné jazyky môžeme zaradiť jazyk PostScript a jeho postscriptové písmo. Od popisu fontu sa vyžaduje, aby pri rôznych druhoch transformácie (napr. rotácia okolo bodu, zväčšovanie atď.) tvar písmena ostal nezmenený. Z tohto dôvodu rasterizačné systémy používajú na popis znakov kubické alebo aj kvadratické krivky. V minulosti sa však považoval popis dostatočný iba s použitím priamok a oblúkov kružníc.

Kubické splajny sú po častiach popísané polynomickými parametrickými krivkami. Od čias čo sú schopné generovnať hladké obrysy, ich používame na popis *hraníc* znakov a na popis postscriptového písma. Prirodzené a obrysové (clamped) splajny sú dané dvoma extrémami a $n-2$ prostrednými bodmi splajnu. Rozličné segmenty splajnu majú spojitosť druhého stupňa a sú teda hladko prepojené v týchto bodoch splajnu. Matematicky je táto hladkosť popísaná ako totožnosť vektorov druhej derivácie v nadväzujúcich bodoch. Pre všetky body spojitosti môžeme zostaviť $n-2$ rovníc. Vyriešením týchto rovníc získame dotykové vektory (dotyčnice) prepojovacích bodov. Každý segment splajnu $P_i(t_i)$ zloženého z dvoch bodov a dotyčníc je možné definovať nasledovne:

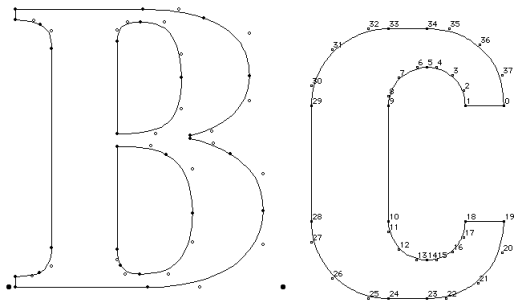
$$\begin{aligned} x_i(t_i) &= a_{xi} + b_{xi} * t + c_{xi} * t_i^2 + d_{xi} * t_i^3 \\ y_i(t_i) &= a_{yi} + b_{yi} * t + c_{yi} * t_i^2 + d_{yi} * t_i^3 \end{aligned} \tag{3.1}$$

Obrázok 3.1: Na ľavej strane ukážka segmentu splajnu a jeho dotyčníc. Na pravej strane ukážka nadväzovania segmentov

Veľmi opatrný treba byť pri parametri t. Na ideálnej krivke by mal byť úmerný dĺžke oblúka. Jeho výber závisí aj od typu zvoleného oblúka. V prípade kruhového oblúka budeme počítať s ideálnym parametrom narozdiel od iného typu oblúka, kde môžeme použiť približný. Týmto sa ušetrí výpočetný výkon.

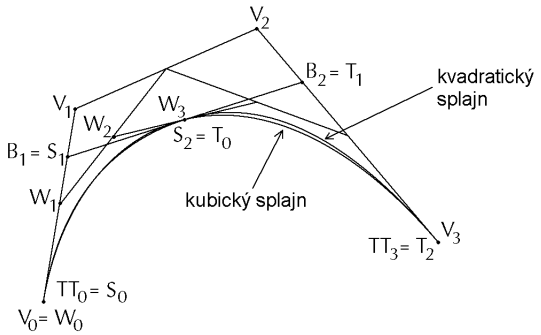
Obrysy znaku sú úplné použitím kubických splajnov a priamok. Pre jednoduchosť pri prenose do rastrovej mriežky a pri vyplňovaní sa vo všeobecnosti konvertujú do ekvivalentnej formy založenej na Bernštajnových polynómoch.

Vo formáte TrueType sú obrysy znakov popísané priamkami a kvadratickými B-splajnami. Tým, že sa používa kvadratický popis je možné znížiť počet bodov a tým aj celkovú veľkosť fontového súboru. Body tejto krivky môžeme rozdeliť do dvoch skupín, a to na body mimo krivky a body patriace krivke. Je možná ľubovolná kombinácia týchto bodov pri definovaní krivky. Body, ktoré vytvárajú krivku musia byť očíslované bod po bode v za sebou idúcom poradí. Zároveň je rozdieľ, či je daná postupnosť rastúca alebo klesajúca. Na základe tejto postupnosti sa neskôr určuje vzor výplne. Smer kriviek musí byť taký, aby bola nasledovaná smerom zväčšujúcich sa čísliel bodov a zároveň čierna vyplnená oblasť musí vždy ležať v pravo.



Obrázok 3.2: TrueType znaky dané bodmi patriacime krivke a bodmi mimo krivku

Na obrázku 3.3 môžeme vidieť, ako je konverzia medzi kvadratickým a kubickým splajnom nedokonalá. Táto konverzia sa používa pri prevode postscriptových písiem do TrueType. Pri hľadaní bodu S_1 sa postupuje pomocou aproximácie, a tá spôsobuje chybu. Nutnosť konverzie odstraňuje až OpenType s dvoma typmi súborov (otf a ttf).



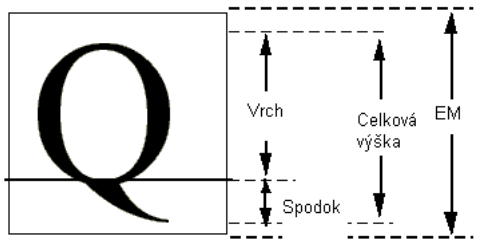
Obrázok 3.3: Konverzia kubického Beziérovho splajnu do kvadratického B-splajnu. Autor: R. D. Hersch

Viac informácií sa dočítate v publikácii [2] alebo na [4].

3.2 Digitalizácia obrysov

3.2.1 Základná jednotka písma

TrueType na popis umiestnenia bodov kriviek a úsečiek používa vo svojich súboroch jednotky písma. Jednotka písma je zároveň najmenšou merateľnou jednotkou využívanou v *em* štvorčeku, ktorý poznáme ako typografickú jednotku pre veľkosť znakov písma. Rozmer *em* štvorčeka je daný celkovou výškou znaku a medzerou navyše, ktorá zabraňuje kolízii s iným riadkom.

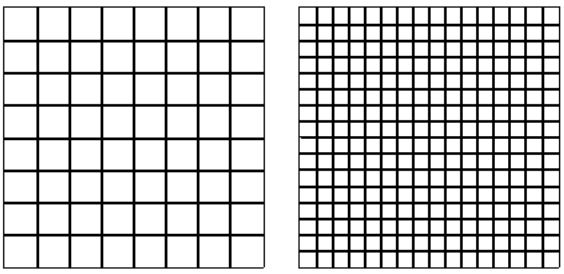


Obrázok 3.4: Veľkosť *em* štvorčeka. Autor: www.microsoft.com

V minulosti sa veľkosť znaku nemohla rozširovať za veľkosť *em* štvorčeka, pretože raziče typografov boli kovové. V dobách digitálnej techniky tento problém odpadá. *Em* štvorček by mal byť dostatočne veľký, tak aby sa do neho pomestil každý znak abecedy, či už s mäkčeňom alebo dĺžňom. V odôvodnených prípadoch je možný presah za štvorček.

3.2.2 Mriežka

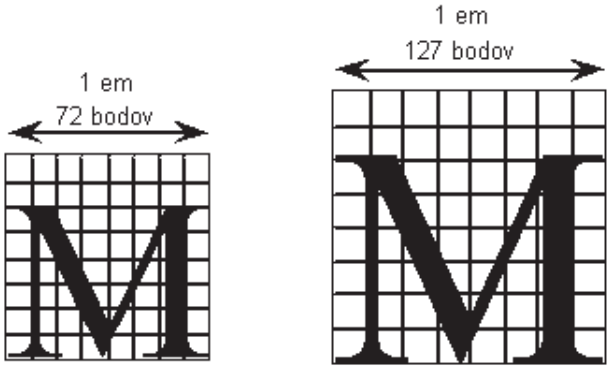
Najdôležitejšia vec pri digitalizácii je zistenie rozlíšenia, do ktorého budú body obrysov znakov vytlačené. Body predstavujú miesta v mriežke, ktorej najmenšia adresovateľná jednotka je jednotka znaku.



Obrázok 3.5: Dva *em* štvorčeky. 8 jednotiek/em (vľavo), 16 jednotiek/em (vpravo). Autor: www.microsoft.com

Každý *em* štvorček je určený počtom základných jednotiek písma. Štvorček rozdelený na tieto jednotky definuje súradný systém, ktorého veľkosť jednej jednotky sa rovná veľkosti základnej jednotky. Všetky body v tomto systéme majú nemenné umiestnenie. Čím viac základných jednotiek pripadá na štorček, tým kvalitnejší výstup môžeme dosiahnuť. Zvyšuje sa nám tým zároveň počet miest, ktoré môžeme adresovať 3.5.

Jednotky písma sú relatívne, pretože ich veľkosť závisí na veľkosti *em* štvorčeka. So zväčšujúcou sa veľkosťou štvorčeka ostáva počet jednotiek nemenný. Nezávisle na veľkosti písma je počet jednotiek taktiež nemenný. Veľkosť bodov na štvorček však závisí od veľkosti písma. Napr. v prípade, že obrys má byť zobrazený na veľkosť 9 bodov, tak aj štvorček bude mať výšku presne 9 bodov. Počet jednotiek písma teda nezávisí na veľkosi písma avšak veľkosť jednotky je závislá od veľkosti písma.



Obrázok 3.6: 72 bodové M a 127 bodové M vo svojich *em* štvorčekoch. Počet jednotiek písma na štvorček je v oboch prípadoch 8. Autor: www.microsoft.com

Relatívnosť jednotiek písma na štvorček prináša zopár výhod. Keď si zoberieme ľubovoľné miesto na znaku, jeho pozícia sa nezmení v závislosti od zmeny veľkosti písma. Obrysy písma je možné spracovať naraz a aplikovať na ňom akékoľvek zmeny bez ohľadu na veľkosť a rozlíšenie kam sa bude rasterizovať.

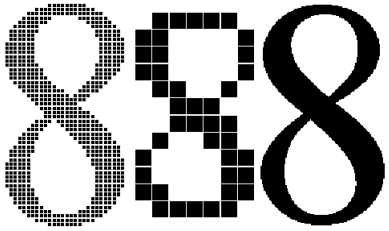
3.2.3 Úprava veľkosti znaku

Nezávisle na rozlíšení štvorčeka, predtým ako môžeme znak zobraziť musí byť prevedený do požadovanej veľkosti, požadovane transformovaný a musí zohľadňovať vlastnosti svojho výstupného zariadenia. Od obrysov je požadované, aby boli reprezentované absolútnymi jednotkami a nie relatívnymi. Pre tento účel sa obrysy popisujú pixelovo.

Konverzia jednotiek písma vo štvorčeku do pixelového súradného systému pri použití merítka veľkosti sa môže riadiť nasledujúcim vzorcom: `veľkosť písma * rozlíšenie dpi na výstupe / (počet bodov na palec * počet jednotiek na štvorček)`.

Ak chceme zistiť presný počet pixelov, ktoré jednotky písma zaberajú, musíme celú našu rovnicu prenásobiť počtom jednotiek, ktoré chceme previezť. Je možné si všimnúť, že v rovnici sa používa delenie, ktoré spôsobuje nutnosť zaokrúhlovať.

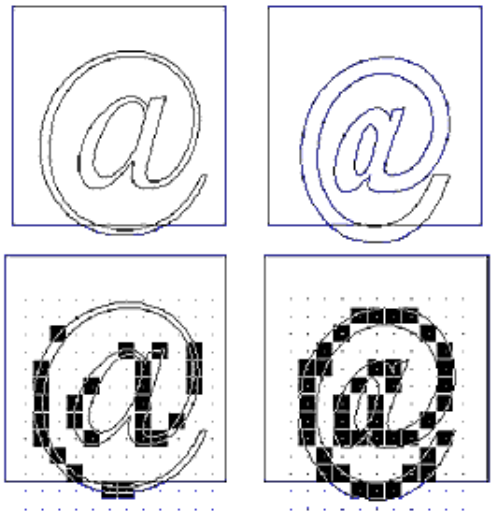
Rozlíšenie každého zobrazovacieho zariadenia je udávané počtom bodov na palec (dpi). Napr. grafická video-mriežka (VGA) používa 96 bodov dpi, laserová tlačiareň 300 bodov dpi. Počet pixelov na štvorček závisí na tomto počte dpi. 18 bodov vysoký znak má pri rozlíšení 72 dpi výšku 18 pixelov. Pri zmene rozlíšenia na 300 dpi bude mať 75 pixelov a pri 1200 dpi bude mať dokonca 300 pixelov na štvorček.



Obrázok 3.7: 18 bodov vysoká '8' pri rozlíšení 300, 72 a 1200 dpi. Autor: www.microsoft.com

3.2.4 Grid-fitting

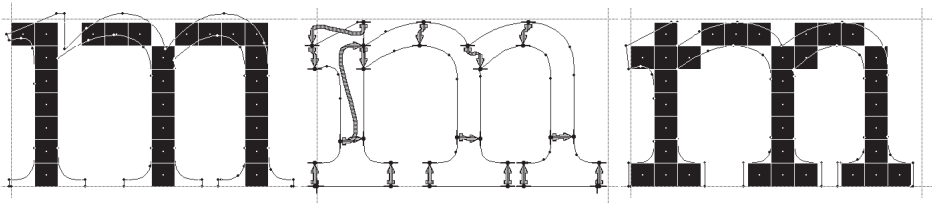
Umiestnenie v súradnicovej sieti patrí medzi najkomplikovanejšie procesy pi rasterizácii. Existujú miesta v popise obrysu znaku, ktoré sú malé, prípadne komplikovane spracované a ich vynechanie môže spôsobiť neželaný výsledok. Myslí sa tým, že pixel ostane nerozsvietený v prípade, že je zaplnený plochou menej ako 50 %. Preto počítač musí byť informovaný v podobe *inštrukcií* ako má daný obrys správne rasterizovať a zarovnať na cielené zariadenie. Ďalej je nutné aby boli vykreslené správne pixely. Pre malé, alebo inak špeciálne typy rozlíšenia sa používajú *hintingové* informácie. Sú to také informácie, ktoré dokážu obrys znaku pokriviť tak, aby sa správne rasterizoval. Celý tento proces sa nazýva *grid-fitting*, čo sa dá voľne preložiť ako zarovnané vykreslenie do mriežky (súradného systému).



Obrázok 3.8: Nezarovnaný (vľavo) a zarovnaný (vpravo) znak zavináč. Autor: www.microsoft.com

Pri rasterizácii TrueType a OpenType fontov sa na zarovnanie používajú inštrukcie. Inštrukcie sú zbierka príkazov, ktorá bola navrhnutá pre dizajnérov, aby dokázali špecifikovať, ako majú byť fonty správne vykreslené. Zároveň sú mechanizmom, ktorý zodpovedá za správne zarovnanie do mriežky. Použitie inštrukcií je možné vypnúť, ale robieva sa to výnimočne na zrýchlenie vykreslovania pri vysokých rozlíšeniach. Pri fonte, ktorý nepoužíva inštrukcie, je výsledok rasterizácie závislí od dizajnu znaku, od veľkosti bodu a od veľkosti rozlíšenia

na výstupnom zariadení.



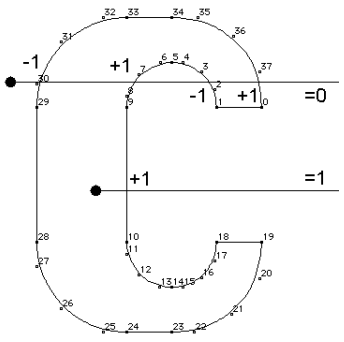
Obrázok 3.9: Obrysy písmena 'm' bez inštrukcií (vľavo) a s inštrukciami (vpravo). Grafická vizualizácia inštrukcií (v strede) Autor: www.microsoft.com

3.2.5 Scan konvertor

Scan konvertor je finálny proces prevodu obrysov do bitmapového obrázku. V predchádzajúcich podkapitolách bolo písané o zarovnávaní do mriežky a s tým súvisiaca určitá deformácia popisu. Miesta vo vnútri obrysu znaku je však nutné aj vyplniť. TrueType používa jednoduchý algoritmus na detekciu, či je bod súčasťou popisu, alebo nie je. Dve základné pravidlá tohto algoritmu:

1. Keď sa stred pixelu nachádza vo vnútri znaku, je rozsvietený a stáva sa súčasťou popisu.
2. Keď niektorá z čiar popisu prechádza stredom pixelu, tento pixel bude rozsvietený.

Body znaku môžeme rozdeliť do dvoch skupín. Body interné a body externé. Bod je interný, keď má nenulové popisné číslo. V opačnom prípade je bod externý. Popisné číslo sa pre každý bod vypočítava nasledujúcim spôsobom: Z bodu namierime lúč smerom na nekonečno (smer nie je podstatný). Počiatočné popisné číslo je nulové. Každý krát, čo lúč prejde niektorou z popisných čiar smerujúcich z pravej strany do ľavej strany, alebo zo spodku na vrch znížime popisné číslo o jednotku. Tento typ kríženia sa nazýva prechod dnu. Každý krát, čo lúč prejde z ľavej strany do pravej strany, alebo z vrchu do spodku popisné číslo o jednotku zväčšíme. Tento typ kríženia sa volá prechod von. Smer kríženia sa určí podľa smeru očíslovania bodov.



Obrázok 3.10: Určovanie popisného čísla

Viac informácií o celej sekcii nájdete na [7] a [6].

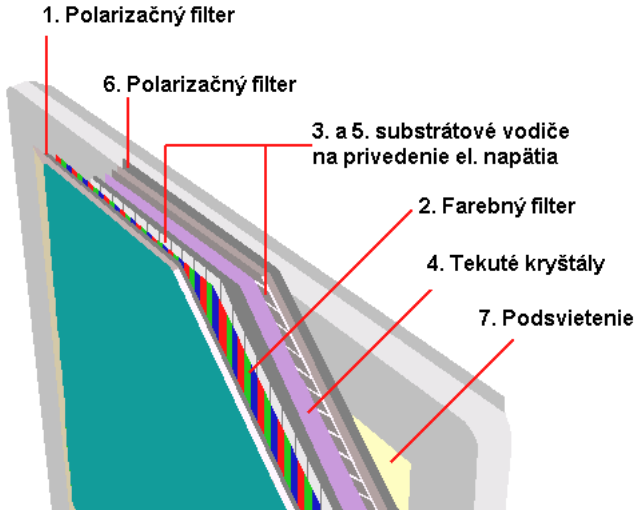
Kapitola 4

LCD

V posledných rokoch popularita LCD obrazoviek rapídne vzrástla. Medzi dôvody jeho popularity môžeme zahrnúť menšiu konštrukciu, nižšiu spotrebu energie ako CRT, ale najdôležitejším aspektom je znížená námaha očí ako pri CRT. V tejto kapitole bude stručne popísaná technológia LCD a jeho skladba pixelu, ktorá je najdôležitejším aspektom pre tvorbu tejto práce.

4.1 Konštrukcia LCD

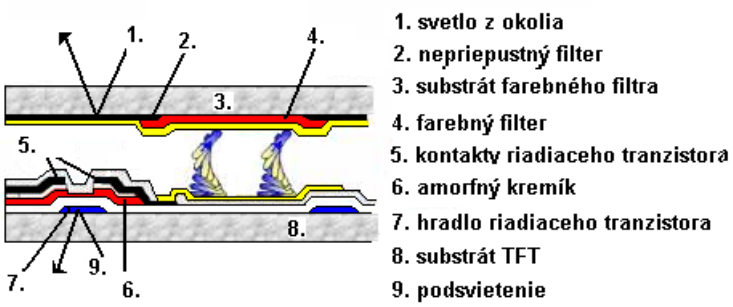
LCD, je skratka, ktorá označuje monitory, ktoré na vytvorenie obrazu používajú mikroskopické kryštály, schopné meniť polohu podľa pôsobenia elektrickej energie. Tieto kryštály majú schopnosť natáčať sa a tým umožniť prechod svetla. Tento princíp je starý už 100 rokov a bol objavený Frederichom Rheinizerom. Prvé LCD displeje sa objavili v kalkulačkách a digitálnych hodinkách. Cena v posledných rokoch výrazne klesla a vďaka tomu LCD panely prekonali v predajnosti CRT monitory.



Obrázok 4.1: Konštrukcia LCD. Autor: www.pctechguide.com

Princíp fungovania TFT LCD je znázornený na obrázku 4.1. Vo väčšine LCD monitoroch sú implementované panely, ktoré používajú na tvorbu obrazu vlastný zdroj svetla.

Tento prvok je na obrázku uvedený ako posledný. Pred ním je umiestnený polarizačný filter, ktorý *nastavuje* svetlo prichádzajúce od zdroja do ideálnej polohy na jeho spracovanie tekutými kryštálmi. Tie sú uzavreté medzi dvoma svetlopriepustnými elektródami (vo väčšine prípadov vyrobených zo skla), na ktoré sa privádza riadiaci signál, ktorý definuje, či má daný prvok svietiť alebo ostať tmavý. Spodná elektróda plní aj úlohu nosiča polovodičových prvkov. Pre každý bod farebného displeja je potrebná trojica tranzistorov na ovládanie elektrického potenciálu, ktorým sú ovládané tekuté kryštály. Horná elektróda má stále rovnaký elektrický potenciál. Je však na nej implementovaný farebný filter, ktorý daný lúč svetla obohatí o farebnú zložku. Ako prvý je polarizačný filter, ktorý je oproti predchádzajúcemu polarizačnému filtru otočený o 90 stupňov. Niekedy plní aj ochrannú funkciu pre celú konštrukciu.



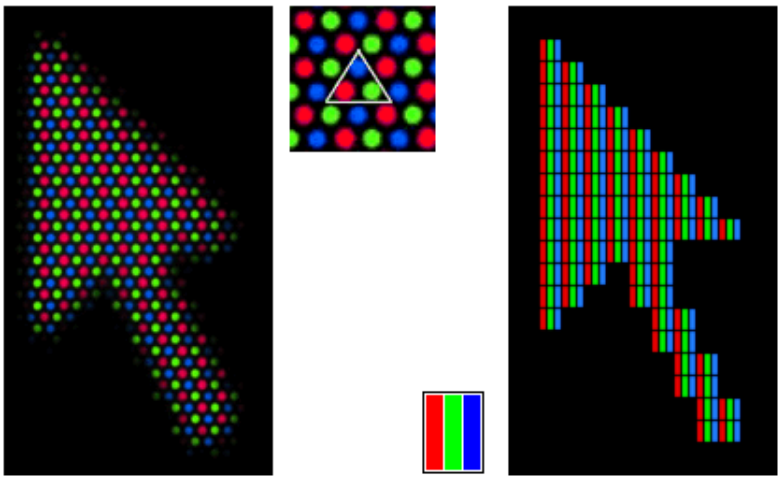
Obrázok 4.2: Konštrukcia LCD. Autor: www.plazma.com

Na obrázku 4.2 je znázornený prvok TFT, ktorý obsahuje komponenty z obrázka 4.1 a navyše sú na tejto schéme viditeľné štruktúry jednotlivých elektród. Hornej dominuje spoločná elektróda a sústava farebných filtrov RGB. Okrem toho tu môžeme nájsť aj pásy neprepúšťajúce svetlo. Na spodnom substráte je vytvorený TFT, ktorý je vyrobený z amorfného kremíka na sklenenej podložke. Jednotlivé elektródy tranzistora sú pripojené k spoločnej zbernici (Source), kontaktu tvoriacemu riadiacu elektródu daného prvku LCD (Drain) a zbernici s riadiacim signálom (Gain). Usporiadanie horného i spodného substrátu zabraňuje prestupu svetla z podsvietenia do štruktúry riadiaceho tranzistora, i jeho prieniku z displeja inou cestou, ako oblasťou, v ktorej sú tekuté kryštály polarizované. Viac informácií nájdete na [5] a [3].

4.2 LCD pixel vs. CRT pixel

Pixel by sme mohli popísať ako najmenšiu jednotku „v počítačovom obrázku", ktorá je schopná zobraziť rôzne farby. Súčasné monitory sú schopné zobrazovať viac ako 16 miliónov farieb. Celý princíp získania týchto farieb spočíva v miešaní intenzít troch základných farieb. Tento princíp je spoločný ako pre LCD tak aj pre CRT. V skutočnosti majú LCD aj CRT pixely 3 virtuálne zložky, a to červenú, zelenú a modrú (RGB). Rozdiel spočíva v usporiadaní týchto zložiek. Pri CRT monitore sú tieto zložky usporiadané do trojuholníka. Pri LCD monitore sú usporiadané vedľa seba. To, že sú usporiadané vedľa seba je pre nás mimoriadne dôležité a túto znalosť využijeme v ďalšej kapitole. V tejto podsekcii sa nachádza aj odpoveď na otázku, prečo sub-pixelové vykreslovanie nie je možné použiť na CRT monitore. Je to zdôvodu nevyhovujúceho usporiadania zložiek do trojuholníka. Navyše po trojuholníku

s červenou zložkou hore musí následovať trojuholník prevrátený.



Obrázok 4.3: Ukážka zložiek pixelov CRT (vľavo) a LCD (vpravo). Autor: www.proline.pl

Sub-pixely a ľudské oko

Všetky 3 farebné zložky sa pri technológii LCD nazývajú sub-pixely. Tieto sub-pixely nie sú pri pohľade zo štandardnej vzdialenosti viditeľné pre ľudské oko. Keď všetky tri zložky svietia na plné intenzity, vytvárajú bielu farbu. V prípade, že ani jednou zložkou neprechádza svetlo, výsledný pixel je čierny. Keďže tieto sub-pixely sú malé, tak ľudské oko si nevšimne, že svieti iba jedna zložka a rozptýli ju do najbližšieho okolia. V prípade, že svietia dve zložky, ľudské oko si samo farby zmieša. Ako už bolo spomenuté, farby sa získavajú pomocou rozsvecovania zložiek na určité intenzity. Keď budeme uvažovať 16 miliónov farieb, tak pre každú zložku máme 255 intenzít (24 bitov / 3 zložky).

Obrázok 4.4: Sub-pixelový riadok (dole) a jeho výsledné farby pixelov na LCD (hore)

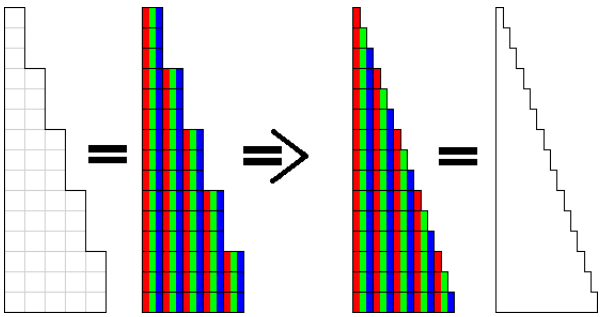
Kapitola 5

Realizácia

Čítaním tejto kapitoly sa oboznámite s postupom práce pri vytváraní demonštračného programu na vykreslovanie fontov s využitím znalostí o sub-pixeloch. Táto kapitola neobsahuje skoro so žiadny programový kód, publikuje hlavne teoreticko-praktický postup. Popri tomto postupe sú predostreté problémy a možnosti ich riešenia. Ďalej je možné oboznámiť sa so sub-pixelovým zobrazovaním, ktoré vzniklo z dôvodu lepšej čitateľnosti fontov na zariadeniach s LCD displejom.

5.1 Rozlíšenie pomocou sub-pixelov

Vďaka sub-pixelom je možné rozlíšenie ,,virtuálne" s trojnásobiť. Ako už vieme z kapitoly o rasterizácii, tak platí, čím väčšie rozlíšenie, tým menšie problémy pri vykreslovaní. Vďaka väčšiemu rozlíšeniu je teda možné dosiahnuť ľubovolný obrys hladší. Okrem toho dokážeme rasterizovať aj veľmi malé písmo, ktoré by v prípade použitia pôvodného rozlíšenia nebolo možné. Bohužial rozlíšenie sa zmení iba v smere, v akom sú orientované zložky pixelov. Podstatným faktom ostáva hladšia rasterizácia, ktorú demonštruje obrázok 5.1.

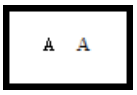


Obrázok 5.1: Tronásobné rozlíšenie a jeho vplyv na hladšie hrany

Na obrázku 5.1 je znázornená rasterizácia okrajov malého trojuholníka. Pri veľkom zväčšení je možné vidieť skokovitý postup pri vykreslovaní prepony trojuholníka. Vďaka maximálnemu využitiu sub-pixelovej rasterizácie môže mať ten istý trojuholník oveľa hladšiu preponu.

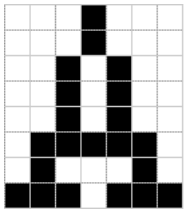
5.2 Čitateľnosť malého písma

Pre veľmi malú veľkosť písma na zariadeniach s malým rozlíšením musí rasterizátor obrysy písmen veľmi deformovať a zdokonalovacie techniky ako hinting mu v niektorých prípadoch nepomôžu. Výsledok nemusi byť podľa očakávaní. Najviac sú takýmto nedokonalým vykreslením postihnuté displeje PDA s malými rozmermi, ktoré si pre pohodlnosť čítania nemôžu dovoliť zväčšiť písmo. V prípade, že je písmo zväčšené, narážame na nižšiu rýchlosť čítania, pretože daný užívateľ sa musí častejšie zaoberať skrolovaním. Na obrázku 5.2 je znázornené písmeno 'A', ktoré trpí spomínaným nedostatkom. Na pravej strane obrázku je znázornené, ako písmeno môže vyzerať s použitím sub-pixelov.



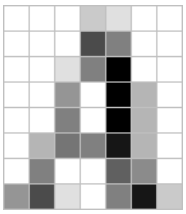
Obrázok 5.2: Písmeno 'A' pri nízkom rozlíšení

Keď sa na obrázok 5.2 pozrieme na pixelovej úrovni 5.3, uvidíme detailne problém nedostatočného rozlíšenia. Jeho pätky nôh sú neprimerane veľké oproti zbytku písmena.



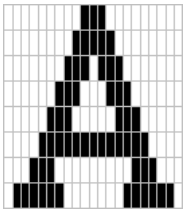
Obrázok 5.3: Zväčšené písmeno 'A' pri nízkom rozlíšení

Prvým záchrancom nedostatku rozlíšenia sa stal anti-aliasing. Pomocou rozptýlených odtieňov šedej farby sa pokúša zobraziť hladšie okraje písmen. Spolieha sa pritom na ľudský zrak, ktorý spriemeruje farbu dvoch šedých susedných pixelov a snaží sa ich vidieť ako jeden pixel s výslednou farbou. Táto technika však nie je vhodná pre malú veľkosť písma, lebo ho rozmazáva. Použitie anti-aliasingu pri malých znakoch núti oči zaostrovať, čo vedie k rýchlej únave očí.



Obrázok 5.4: Zväčšené písmeno 'A' pri nízkom rozlíšení a použití anti-aliasingu

Najlepšou odpoveďou na nízke rozlíšenie je možnosť použiť sub-pixely. Písmeno tak môže mať hladšie čiary oproti klasickej skokovej úsečke. Priestor sa dostáva aj na pätky písmena A, ktoré môžu byť zobrazené na veľkosť určenú popisom.



Obrázok 5.5: Zväčšené písmeno 'A' pri nízkom rozlíšení a použití sub-pixelov

5.3 Použitie a výhody sub-pixelov

Použitie sub-pixelov prináša výhody nielen na poli vyhladzovania hrán, ale aj pri umiestňovaní obrysov pomocou *kerningu*. *Kerning* si môžeme predstaviť ako spôsob určovania veľkosti medzier medzi jednotlivými znakmi. Najlepším príkladom je obrázok 5.6, v ktorom sú zobrazené písmená 'V' a 'A'. Obidve písmená môžu byť rasterizované samostatne ako *em* štvorček alebo vďaka kerningu sa môžu tieto štvorčeky prekrývať. Keďže *kerning* dokáže zvyšovať rýchlosť čítania textu, patrí do štandardnej zbierky popisu fontov.

Obrázok 5.6: VLTAVA bez *kerningu*(hore) a s *kerningom*(dole)

Výhoda kerningu sa najviac prejavuje pri malých veľkostiach písma. Pixel je do pomeru s fontom neúmerne veľký. Preto kerning nie je možné aplikovať bez použitia sub-pixelov.

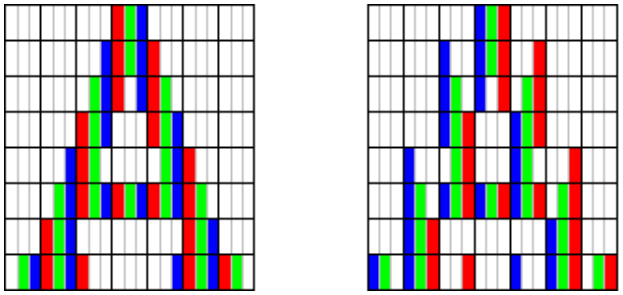
Ďalšou výhodou sub-pixelov je zobrazovanie tučného písma. Opäť pri prípade malého písma sa môže tučné písmo zdať príliš tučné. V iných prípadoch môže vzbudiť dojem nízkej tučnosti. Pomocou sub-pixelov je možné znížiť vplyv týchto nedostatkov.

Kurzíva sa používa na zatraktívnenie a zvýraznenie pasáže v texte. Dokáže nerušene zaujať čítajúceho viac ako v prípade tučného písma. Takisto ako v prípade tučného písma je v prípade malého písma kurzíva príliš sklonená, alebo jej sklon je nedostatočný. Sub-pixely pomocou vyhladzovania hrán sú nápomocné aj pri vernejšom zobrazení obrysov.

5.4 RGB vs. BGR

Tieto tri písmená predstavujú usporiadanie základných troch farieb. V minulosti sa vyrábali LCD panely s BGR usporiadaním sub-pixelov. Keďže našou snahou je adresovať dané sub-pixely, musíme byť oboznámení s ich usporiadaním. Dôležitosť znalosti tohto usporiadania demonštruje obrázok 5.5. Je v ňom znázornené používanie adresácie RGB sub-pixelov pri LCD s rovnakým usporiadaním pixelov a LCD s opačným usporiadaním pixelov. Pri pohľade na vnútorné pätky ľavého písmena 'A' vidíme červený a modrý pixel nalepený k ostatným sub-pixelom. Keď sa však snažíme adresovať tie isté sub-pixely, tak červený aj modrý sub-pixel sa od kontúr písmena 'A' prevrátia na opačnú stranu. Nič sa však nezmení, keď zanedbáme existenciu zložiek pixela a berieme do úvahy len výslednú farbu pixela. Tento fakt

je opäť odpoveďou na otázku, prečo nie je možné tento benefit použiť na CRT monitore, aj keď sa zdá, že na ňom dosahuje zlepšenie. Pre správnu funkciu nášho algoritmu 5.5.6 musíme byť informovaný akým spôsobom je potrebné sub-pixely adresovať.



Obrázok 5.7: Adresovanie RGB sub-pixelov na LCD type RGB (vľavo) a BGR(vpravo)

Okrem spomenutých usporiadaní pixelov existujú aj vertikálne usporiadania s označením vRGB a vBGR.

5.5 Postup pri tvorbe programu

V predchádzajúcich kapitolách bolo popísaných množstvo teoreticko-praktických znalostí, ktoré väčšinou súvisia s realizáciou demonštračného programu. Priebeh práce je popísaný v následujúcich podkapitolách.

5.5.1 Adresácia sub-pixelov

Prvoradým cieľom je premyslieť prácu s jednotlivými zložkami pixelu a ako sa k nim dopracovať. Možnosť pristupovať k sub-pixelom na hardwarovej úrovni je veľmi zložitá. Pre nás je prístupné jedine adresovanie na úrovni pixelov. Existuje ale riešenie, pri ktorom je možné so sub-pixelmi pracovať veľmi jednoducho a efektívne bez nutnosti adresácie na úrovni hardwaru. Táto možnosť vychádza z teórie pixelovej farby na LCD.

Farba	Sub-pixle
Červená	R
Zelená	G
Modrá	B
Žltá	RG
Cyan	GB
Magenta	RB
Biela	RGB
Čierna	

Obrázok 5.8: Adresovanie RGB sub-pixelov pomocou farieb

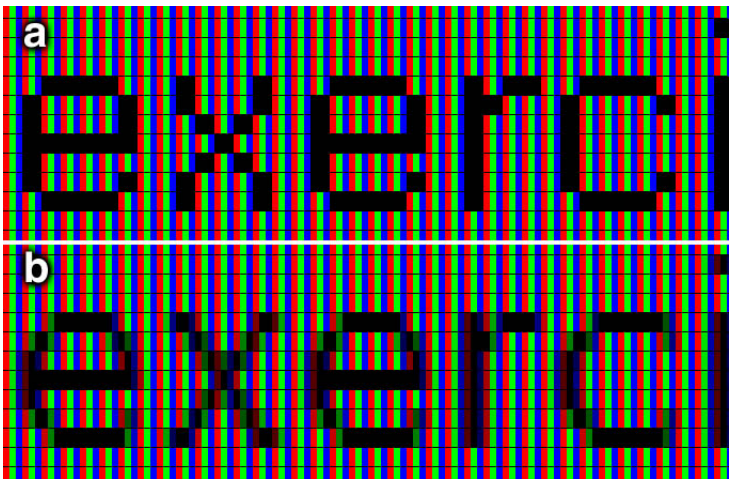
Keď chceme vysvietiť červený sub-pixel, tak jednoducho dáme vykresliť červenú farbu pre celý pixel. Máme zároveň zaručené, že všetky ostatné zložky zostanú nerozsvietené. Zároveň prvý sub-pixel za dvoma zhasnutými zložkami je červený sub-pixel. Ľudské oko je tak nedokonalé, že si červenú farbu ,,rozleje" do celej oblasti a ignoruje zhasnuté sub-pixely. Rovnako je možné postupovať aj v prípade že chceme adresovať inú kombináciu zložiek pixelu.

Vlastná tvorba

Fakty, ktoré boli uvedené doteraz, je možné nájsť v literatúre. Nikde však nenájdeme ako presne na to. V nasledujúcich stranách sa môžete dočítať o mojom vlastnom postupe pri implementácii sub-pixelového vykreslovania.

5.5.2 Spôsob rasterizácie

Najväčším problémom pri úvahách o tvorbe programu bol druh rasterizácie. Nie je možné použiť rasterizáciu do rozlíšenia na obrazovke. Stratili by sa tak dokonalejšie hrany, ktoré sub-pixely dokážu vytvárať. Ani následná editácia už rasterizovaného fontu by nedokázala priniesť tak dokonalé hrany ako v prípade trojnásobného rozlíšenia. Je teda potrebné nájsť funkciu, ktorá dokáže rasterizovať na úrovni sub-pixelov a tým dokáže font obohatiť o hladšie krivky. Všetky nájdené funkcie už ponúkali svoju vlastnú implementáciu sub-pixelového zobrazovania. Najznámejšou z nich je ClearType od Microsoftu, ktorý je vo Windows Vista už vo svojej druhej verzii a pridáva rôzne vylepšenia.



Obrázok 5.9: Rasterizovaný text pomocou Microsoft ClearType. Autor: www.wikipedia.com

Keďže žiadna funkcia pre priamu rasterizáciu nie je použiteľná, tak je nutné improvizovať. Z teórie vieme, že sub-pixely nám virtuálne strojnásobia rozlíšenie. Odpoveďou na náš problém je funkcia, ktorá dokáže rasterizovať text do trojnásobnej šírky pri zachovaní výšky písma. Implementačný jazyk teda priamo závisí od funkcie, ktorá v ňom musí existovať. Najväčším problémom pri vyhľadávaní bola predchádzajúca nesprávna úvaha. Vykreslovanie fontu do danej šírky totiž nezabezpečujú funkcie rodiny ,,kresli text". Šírku je potrebné definovať už v počiatku vykreslovania–definícii fontu.

Pri hľadaní som nenarazil na žiadnu funkciu platformy .NET, ktorá by umožňovala rasterizovať text do trojnásobnej šírky. Najbližšou možnosťou pri definovaní typu fontov bolo použitie ClearType. Od vzniku ClearType sa Microsoft silno drží svojej implementácie a neumožňuje iný spôsob vlastnej realizácie tejto technológie. Starším rasterizátorom od Microsoftu je GDI. GDI je rasterizátor zabudovaný v jadre operačného systému Windows a pristupuje sa k nemu pomocou WinApi funkcií. Vo WinApi existuje funkcia ,,HFONT CreateFont", ktorá dokáže text rasterizovať do rôznej výšky a šírky. Na obrázku 5.10 je možné vidieť parametre tejto funkcie a ich komentáre. Druhý parameter tejto funkcie definuje *priemernú* šírku znaku zvoleného typu fontu. Bohužial priemerná nie je presne trojnásobná, ale aj napriek danému hendikepu je funkcia ,,CreateFont" prvým kandidátom na použitie v programe.

```
HFONT CreateFont(
  int nHeight,             // height of font
  int nWidth,              // average character width
  int nEscapement,         // angle of escapement
  int nOrientation,        // base-line orientation angle
  int fnWeight,            // font weight
  DWORD fdwItalic,         // italic attribute option
  DWORD fdwUnderline,      // underline attribute option
  DWORD fdwStrikeOut,      // strikeout attribute option
  DWORD fdwCharSet,        // character set identifier
  DWORD fdwOutputPrecision,  // output precision
  DWORD fdwClipPrecision,    // clipping precision
  DWORD fdwQuality,        // output quality
  DWORD fdwPitchAndFamily,   // pitch and family
  LPCTSTR lpszFace         // typeface name
);
```

Obrázok 5.10: Parametre funkcie CreateFont

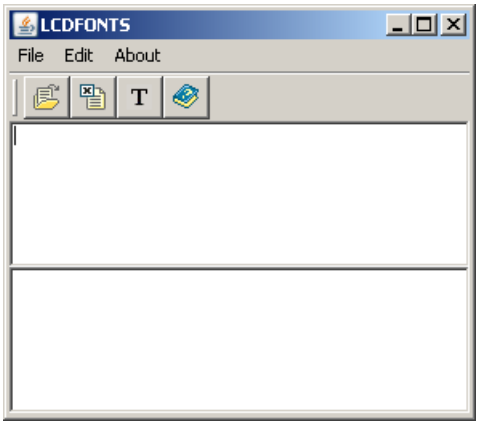
Freetype je voľne šíriteľná knižnica napísaná v jazyku C, ktorá poskytuje možnosť rasterizácie fontov. Túto knižnicu používa väčšina Linuxov na rasterizáciu textu. Taktiež obsahuje vlastnú realizáciu sub-pixelovej rasterizácie na LCD. Použitie jej funkcií FT_Glyph_Metrics na zistenie šírky každého znaku a FT_Set_Char_Size na definovanie jeho novej šírky by bolo mimoriadne pracné. Pri vykreslovaní je navyše nutné napoziciovať každý znak ručne.

Mimoriadne vhodným by prišlo použitie jazyka JAVA, ktorý je multi-platformový a má v sebe zabudovaný vlastný rasterizátor. V triede ,,TextAttribute" je možné pomocou metódy Width nastaviť šírku daného fontu. JAVA sa tak vďaka jednoduchosti, použitiu vysokej formy abstrakcie a v neposlednom rade možnosťou nastavovať šírku celého fontu stáva zvoleným implementačným jazykom.

Jednou zo skúmaných možností bolo použitie trojnásobnej výšky fontu. Vďaka tomu by sa nám zväčšila aj šírka písma. Následným použitím translácie na výšku by sme mohli dosiahnuť podobný efekt ako v prípade použitia triedy TextAttribute. Narážame ale na problém vylepšenia rasterizácie vektorových fontov ,,smart outlines", vďaka ktorému je možné zmeniť tvar kriviek pri zmene veľkosti písma. V tomto prípade by však mohol byť zvolený ľubovolný implementačný jazyk. Ďalšou nevýhodou je nutnosť mať 3-násobne väčšie plátno (z dôvodu výšky) ako pri všetkých ostatných metódach.

5.5.3 Vzhľad a možnosti aplikácie

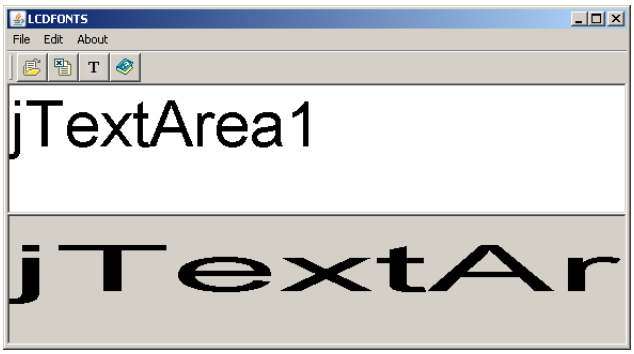
Cieľom aplikácie je demonštrovať rozdiely medzi rôznymi druhmi ,,typeface" rezov (konkrétny typ písma s presnou veľkosťou a atribútmi). V hornej časti je umiestnené textové pole do ktorého je možné písať a editovať text. V spodnej časti programového okna je pre užívateľa needitovateľná bitmapa. Medzi plánované funkcie patrí otváranie súborov, ukladanie súborov, zmena rezu písma a zoom.



Obrázok 5.11: Vzhľad demonstračnej aplikácie

5.5.4 Rasterizácia textu

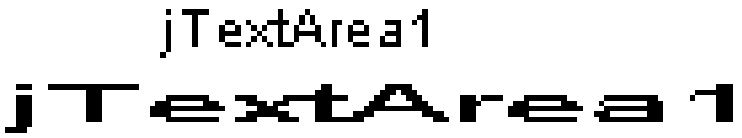
Plátno, do ktorého ideme kresliť, musí mať 3-násobnú šírku oproti plátnu, ktoré je definované pomocou textovej oblasti. Týmto si zaručíme, že naše písmo sa rasterizuje celé. Prvým cieľom je vykresliť určitý text do trojnásobnej šírky. Výsledok demonštruje obrázok 5.12.



Obrázok 5.12: Rasterizácia trojnásobnej šírky písma

Pri pohľade na obrázok 5.13 vytvorený z obrázku 5.12, je možné vidieť prvé zlepšenie pri rasterizácii. Na texte, ktorý je rasterizovaný ako prvý (vyššie), je možné vidieť vplyv nedostatočného rozlíšenia. Najviac týmto nedostatkom trpí malé písmeno 'a'. Text, ktorý bol vykreslený na plátno (nižšie) dosahuje omnoho lepšie vlastnosti. Malé písmeno 'a' je rasterizované presnejšie. Ďalej je možné si všimnúť veľké písmeno 'A'. Pri rasterizácii do ,,normálnej" šírky sú jeho zvislé hrany príliš skokové. Naproti tomu funkcia, ktorá nám

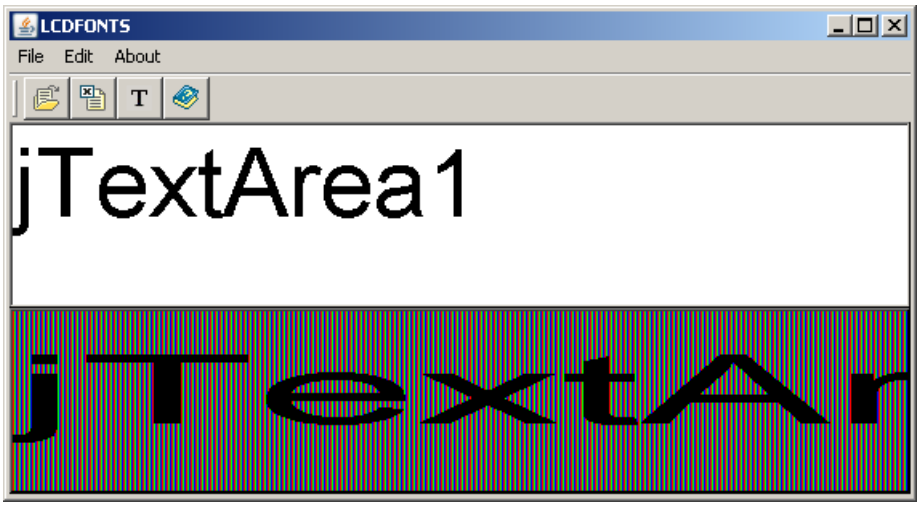
rasterizovala text do 3-násobnej šírky, priniesla písmenu 'A' detailnejší popis. Tento popis textu je síce taktiež skokový, ale pri spätnej transformácii do ,,klasickej" šírky s použitím sub-pixelovej technológie bude tento text zobrazený s hladšími hranami a presnejšie.



Obrázok 5.13: Porovnanie dvoch rasterizovaných fontov

5.5.5 Prevod plátna do RGB

Náš špeciálny reťazec je úspešne vykreslený na obrazovke. Ďalším krokom je prispôsobiť vzhľad plátna do podoby na monitore LCD. Jednoducho povedané algoritmom každý pixel prekonvertovať na farby, z ktorých sa skladá LCD. Vieme, že každý svietiaci pixel na LCD používa plné hodnoty sub-pixelov RGB. Keď v plátne demonštračného programu rátame od ľavého okraja, tak vždy začíname celým, červeným pixelom. Keďže zložky sú tri, tak je potrebné skontrolovať aj ďalšie dva susedné pixely. Ak je susedný pixel biely, tak jeho farba sa prekonvertuje na zelenú. Ak je ,,sused suseda" biely, jeho farba sa prekonvertuje na modrú. V prípade, že pixel je vyfarbený na čierno (nevysvietený je tým pádom aj na LCD) jeho farba ostáva nezmenená. Intenzita tohto sub-pixelu je zároveň nulová. Týmto spôsobom sa prekonvertuje celé plátno do podoby aká je na obrázku 5.14.

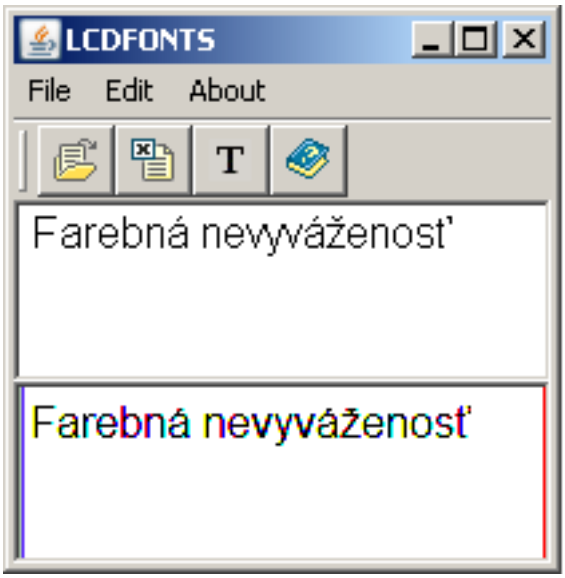


Obrázok 5.14: Plátno transformované do RGB podoby

5.5.6 Zobrazenie výsledného pixelu

Plátno plné RGB pixelov rovnakých ako skupina sub-pixelov na LCD je nutné prekonvertovať do normálnej šírky. Pre každé tri pixely pôvodného plátna vytvoríme jeden nový.

Jeho výsledná farba bude rovná farbe zmiešanej z troch starých pixelov. Opäť je nutné začať od okraja a popísaný algoritmus aplikovať na skupiny troch pixelov. Na obrázku 5.15 je realizovaný náš prevod. Bohužiaľ, stretávame sa s prvým problémom, ktorý sa nazýva farebná nevyváženosť. Na obrázku 5.15 je možné vidieť písmeno 'F', ktoré dostalo na ľavej strane žltý ,,nádych". U všetkých ostatných písmen je možné spozorovať podobný problém.



Obrázok 5.15: Ukážka farebnej nevyváženosti

Farebná nevyváženosť

Najlepšou formou vysvetlenia prečo problém farebnej nevyváženosti vzniká je pohľad na úrovni sub-pixelov. Tento pohľad je znázornený na obrázku 5.16.

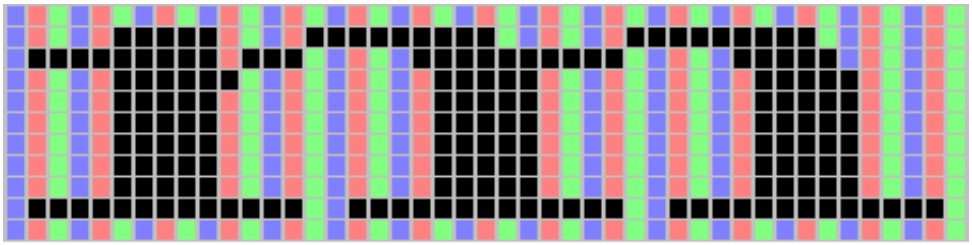
Pri pohľade na nožičky písmena 'm' uvidíme, že sú z oboch strán ohraničené červenými sub-pixelmi. Nožičky sa teda rasterizovali na šírku 5 pixelov. Keďže rasterizátor zaokrúhľuje, v klasickej forme rasterizácie by každá z nožičiek bola vykreslená na šírku dvoch pixelov. Aby sme dosiahli najlepší farebný výsledok mala by šírka byť deliteľná tromi. Pri sub-pixelovej rasterizácii nám zostáva jedna zložka pixelu rozsvietená, ktorá spôsobí, že druhý pixel získa farbu odlišnú od čiernej.

Farebná nevyváženosť sa dá vysvetliť aj následovne: nožičky písmena 'm' majú nevysvietené G-B-R-G-B sub-pixely. V tejto postupnosti sú dve zelené, dve modré, ale iba jedna červená. To znamená, že v prvom pixeli svieti iba červená zložka. Naše oči si túto farbu rozložia do celého pixelu. Celkovo táto ,,kolorita" pôsobí mimoriadne rušivo pre ľudský zrak.

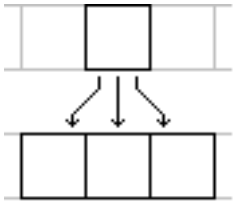
Normálny druh rasterizácie nemá žiadny problém s farebnou nevyváženosťou pretože na rasterizáciu používa všetky zložky pixelu, resp. celý pixel.

Riešenie farebnej nevyváženosti

Našu stratégiu je nutné mierne upraviť. Riešením je distribúcia energie sub-pixelu do jeho bezprostredného horizontálneho okolia.



Obrázok 5.16: Písmeno 'm' v RGB mriežke. Autor: www.grc.com



Obrázok 5.17: Susedné pixely. Autor: www.grc.com

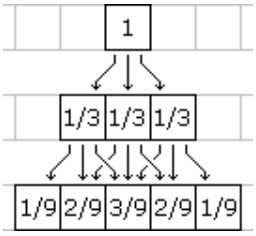
Cieľom je nastaviť intenzitu svietenia konkrétneho sub-pixela tak, aby sme odstránili farebnú nevyváženosť. Postup ako sa ,,zbaviť kolority" je možné rozdeliť do niekoľkých krokov. V prvom rade je nutné si pamätať všetky svietiace sub-pixely. Keď znížime intenzitu všetkých rozsvietených sub-pixelov na nulu, všetky sa stanú zhasnutými. Pre každý pôvodne svietiaci sub-pixel aplikujeme nasledovný filter: Ak bol sub-pixel pôvodne rozsvietený, tak mu pridaj 1/3 celkovej intenzity. Taktiež pridaj 1/3 celkovej intenzity jeho bezprostredným susedom. Keď sa nachádzajú veďľa seba tri svietiace pixely, tak stredný z nich bude svietiť na plnú intenzitu tj. 3/3, krajné budú svietiť na 2/3 intenzitu a susedné zhasnuté pixely sa rozsvietia na 1/3 intenzity. Princíp je demonštrovaný na obrázku 5.20. Tento spôsob obsahuje prvky anti-aliasing, kde okolie pixelu sa ,,rozmazáva", a preto je možné tento spôsob považovať za farebný anti-aliasing.

farebná nevyváženosť

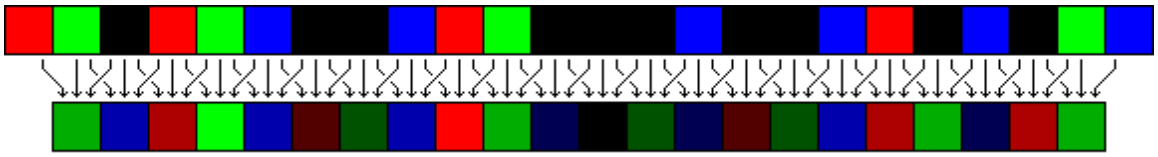
Obrázok 5.18: Ukážka textu pri použití prenosu 1/3 energie bezprostredným susedom

Prenos 1/3 energie stále spôsobuje miernu farebnú nevyváženosť. Preto je vhodné energiu ,,rozbiť" na menšie kúsky a distribuovať ju na väčšiu vzdialenosť. Na obrázku 5.19 je znázornené ako rozsvietený koreňový sub-pixel daruje 1/3 energie sebe a 1/3 energie svojim susedom. Tento proces je však aplikovaný v dvoch fázach s použitím energie koreňového sub-pixelu. Výsledkom je, že pôvodný sub-pixel preniesol 1/9 energie vzdialeným susedom, 2/9 energie bezprostredným susedom a 3/9 samému sebe. Prenos energie na ďaľšie vzdialené sub-pixely už nie je vhodný.

Súčet darovanej energie v poslednom riadku na obrázku 5.19 je rovný jednej. Sub-pixel teda daroval všetku svoju energiu. Obidva popísané spôsoby distribuujú najviac energie svojim koreňovým sub-pixelom. V opačnom prípade by výsledný pixel mohol pôsobiť dojmom, že nepatrí k okolitým pixelom. Viac informácii o tomto algoritme nájdete na [1].



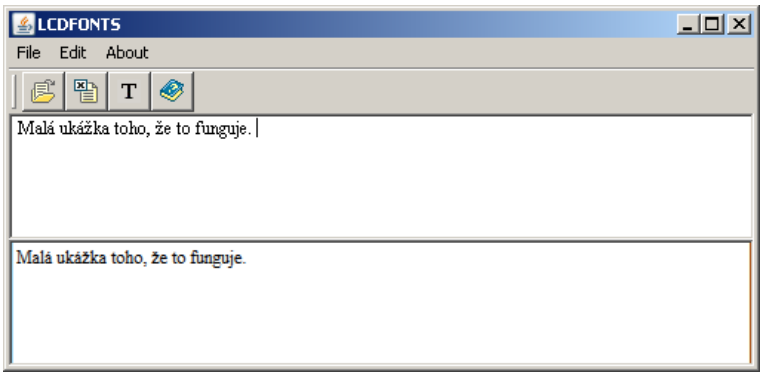
Obrázok 5.19: Prenos energie viacerým susedom. Autor: www.grc.com



Obrázok 5.20: Prenos energie viacerým susedom. Autor: www.grc.com

5.5.7 Použitie filtra

Pomocou algoritmu 5.5.6 je možné vytvoriť filter, ktorý nám odstráni farebnú nevyváženosť. Tento filter nie je možné aplikovať na súčasné plátno, pretože výsledný pixel už je zobrazený. Preto tento filter treba použiť pred zobrazením výslednej farby pixelu zhotovenej zo sub-pixelov. Výstupom z filtru je pole hodnôt, ktoré predstavujú intenzity jednotlivých sub-pixelov. Každé tri za sebou idúce hodnoty poľa predstavujú intezity farieb RGB. Z pôvodných 2^3 farieb môže teraz vzniknúť 9^3 farieb. Výsledok použitia filtru je vidieť na obrázku 5.21.

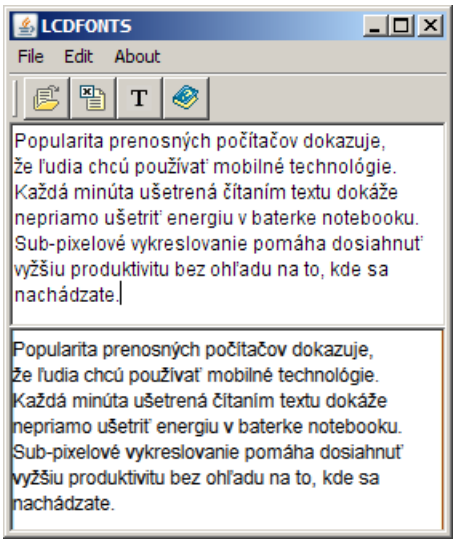


Obrázok 5.21: Text rasterizovaný bez znalosti sub-pixelov(hore) a so znalosťou sub-pixelov (dole)

5.6 Porovnanie výsledkov

Technológia vykreslovania pomocou sub-pixelov je známa už viac ako 20 rokov. S týmto princípom je možné sa stretnúť pod názvom ClearType v operačnom systéme Windows

XP a Windows Vista. Microsoft vidí v zlepšovaní čitateľnosti textu veľký potenciál. Štúdie dokazujú, že ClearType dokáže zrýchliť rýchlosť čítania o 5 %. Je to síce malé číslo, ale keď sa pozrieme na zrýchlenie z manažérskeho hľadiska, tak zistíme, že je možné ušetriť až 3 minúty za jednu hodinu.



Obrázok 5.22: ClearType text (hore) a sub-pixelový text (dole)

Presná implementácia ClearType nie je zverejnená, preto nie je možné dosiahnuť rovnaké výsledky. Na obrázku 5.22 je znázornené porovnanie obidvoch rasterizácii. Text vykreslený pomocou technológie ClearType používa iný rasterizátor ako JAVA. Preto je možné si všimnúť rozdielne dĺžky riadkov. Toto je jediný nedostatok ClearType, ktorý dokáže text vykreslovať len od začiatku celého pixelu. ClearType je ale možné použiť na ľubovolnú farbu textu a na ľubovolnom pozadí. Microsoft investoval do ClearType veľké množstvo peňazí, preto porovnanie kvality zobrazenia alebo porovnanie rýchlosti čítania textu nie je opodstatnené.

Kapitola 6

Záver

Teória okolo písma by vyžadovala oveľa viac stránok ako má táto bakalárska práca. Síce to z názvu nevyplýva, ale opäť zdôrazím, že podtitulom práce by mohol byť *čitateľnosť písma*. Vďaka čitateľnosti vznikol celý rozruch okolo sub-pixelového zobrazenia. Nie je možné byť stručný na úrovni heslovitých poučiek. V takomto prípade nám môže uniknúť množstvo zaujímavých súvislostí. Preto nestačí povedať jednu vetu o existencii a použití vektorových fontoch, ale treba ich aj porovnať s bitmapovými. Forma popisu fontov sa už dlhé roky nezmenila, pretože neexistuje na svete ďalší gigant typu Microsoft alebo Adobe, ktorý by si vedel túto zmenu vynútiť. Na druhej strane treba uznať, že nová forma popisu nie je nutná, pretože počas existenicie OpenType sa významne nezmenili zobrazovacie zariadenia.

Rasterizácia písma je proces, ktorý mnohým počítačovo znalým odborníkom nič nehovorí. Väčšina takýchto ľudí sa stretáva len s rasterizáciou základných grafických primitív ako sú úsečka alebo elipsa. Táto znalosť pramení z jednoduchosti použitých algoritmov, ale aj možnosti stretnúť sa s danými grafickými útvarmi počas štúdia na strednej škole. Naproti tomu teória kriviek ostáva zahalená tajomstvom. Ešte horšie je to v prípade prvého stretnutia sa s rasterizáciou kriviek, ktorej rozumie veľmi málo študentov.

Táto práca sa nepokúsila obzrejmiť algoritmy rasterizácie grafických primitív a kriviek používaných pri rasterizácii písma. Je v nej však použitá teória v kombinácii s praxou. Nie je potrebné poznať kompletné detaily rasterizácie. Stačí poznať postup práce vo vyššej forme abstrakcie. Kapitola o rasterizácii toto dokonale splňuje.

V maximálnej stručnosti je možné oboznámiť sa s technológiou LCD a jej hlavným prínosom pre nás – v jej usporiadaní pixelov.

Realizácia môže zo začiatku pôsobiť teoretickým dojmom. Je to spôsobené komplexnosťou okolo sub-pixelového zobrazenia. Podľa môjho názoru je vhodné pri spôsobe realizácie zároveň popísať o všetkom čo môže byť priamo užitočné. Nepriamo užitočné súvislosti boli popísané v predchádzajúcich kapitolách.

Prvým neúspechom pri realizácii bol nesprávne zvolený dátový typ BufferedImage. Tento typ môže alokovať obmedzené množstvo pamäti. Preto v prípadoch, keď je v TextArea veľa textu program vygeneruje výnimku. Z tejto príčiny nie je možné otvárať rozsiahle súbory. Nesprávne zvolený dátový typ alebo aj postup pri programovaní sa odzrkadlil pri realizácii funkcie zoom. Keďže ani jeden z týchto nedostatkov neohrozuje demonštračnú činnosť programu, nepovažoval som za nutné program prerobiť. Zadanie som v tomto prípade taktiež splnil.

Ďalší problém vzniká pri odlišnosti dvoch rasterizátorov. V hornej časti programu je použitý rasterizátor operačného systému. V spodnej časti programu sa využíva rasterizátor jazyka JAVA. Práca týchto dvoch rasterizátorov je podobná, bohužial však nevytvárajú

identické výsledky. Tieto rozdiely sú spôsobené zaokrúhlovaním, ale aj odlišným nastavením auto-hintingu a kerningu. Z tohto dôvodu scrollbar nie je v programe najdokonalejší. Scrollbar vychádza zo šírky znakov, ktoré používa Windows.

V budúcnosti by bolo potrebné vylepšiť prácu s pamäťou. Použitie scrollbaru doviesť k dokonalosti. Ďalším smerom rozvoja by bola sub-pixelová rasterizácia rôznej farby fontu, na rôznom pozadí. Túto funkciu má implementovaný Microsoft Cleartype. Efekt sub-pixelov sa najlepšie prejaví na silne kontrastných farbách akými sú biela a čierna.

Tému, ktorú som si zvolil ma zaujala. Pri študovaní literatúry som sa najviac pobavil na histórii, keď sa vytvárali aliancie spoločností voči iným monopolom. Najdlhšie som pracoval na hľadaní spôsobu, ako rasterizovať text do šírky. Celkovo som s mojou prácou spokojný a neľutujem výber danej témy.

Zoznam použitých zdrojov

[1] S. Gibson. Sub-pixel font rendering technology. [online], [rev. 2003-11-28], [cit. 2007-05-09]. Dostupné na URL: <http://www.grc.com/cleartype>.

[2] R. D. Hersch. Font rasterization: the state of the art. pages 78–109, 1993.

[3] P. Orviský. 14x 15 LCD monitory. *PC REVUE*, 10:54–61, 2002.

[4] L. Penney. TrueType Outlines. 1996. [online], [rev. 2005-11-16], [cit. 2007-05-07]. Dostupné na URL: <http://www.truetype-typography.com/ttoutln.htm>.

[5] WWW stránky. Learn About LCD TV and TFT LCD Displays. [online], [cit. 2007-05-07]. Dostupné na URL: <http://www.plasma.com/classroom/what_is_tft_lcd.htm>.

[6] WWW stránky. The raster tragedy at low resolution. 1997. [online],[rev. 1998-03-25],[cit. 2007-05-07]. Dostupné na URL: <http://www.microsoft.com/typography/tools/trtalr.htm>.

[7] WWW stránky. TrueType fundamentals. 1997. [online], [rev. 1997-11-16], [cit. 2007-05-07]. Dostupné na URL: <http://www.microsoft.com/OpenType/OTSpec/TTCH01.htm>.

[8] K. Třešňák. OpenType. 2001. [online], [rev. 2001-04-23], [cit. 2007-05-07]. Dostupné na URL: <http://www.printing.cz/art/fonty/opentype_uvod.html>.

[9] K. Třešňák. PostScriptové a TrueType fonty. 2001. [online], [rev. 2001-02-20], [cit. 2007-05-07]. Dostupné na URL: <http://www.printing.cz/art/fonty/ps_vs_tt.html>.

[10] Wikipedia. Computer font. 2007. [online], [rev. 2007-04-29], [cit. 2007-05-07]. Dostupné na URL: <http://en.wikipedia.org/w/index.php?title=Computer_font>.

[11] WWW stránky. Font, slovník. [online], [cit. 2007-05-07]. Dostupné na URL: <http://www.zive.sk/slovnik/default.asp?EXPSDIC=font>.

[12] M. Švamberg. Jak na LaTeX: tabulky I. 2001. [online], [rev. 2001-08-29], [cit. 2007-05-07]. Dostupné na URL: <http://www.root.cz/jak-na-latex-tabulky/>.

[13] R. Černý. Co je OpenType?. [online], [cit. 2007-05-07]. Dostupné na URL: <http://www.fontexplorer.cz/static.php?page=co-je-open-type>.

[14] R. Černý. Co je TrueType?. [online], [cit. 2007-05-07]. Dostupné na URL: <http://www.fontexplorer.cz/static.php?page=co-je-true-type>.

[15] R. Černý. PostScript (Type 1). [online], [cit. 2007-05-07]. Dostupné na URL: <http://www.fontexplorer.cz/static.php?page=co-je-postscript>.

Zoznam použitých skratiek a symbolov

OS - Operating System

OTF - OpenType Format

TTF - TrueTepy Format

ATM - Adobe Type Manager

INF - INFormation file

PFM - Printer Font Metric

AFM - Adobe Font Metric

DPI - Dot Per Inch

VGA - Video Graphic Adapter

LCD - Liquid Crystal Display

TFT - Thin Film Transistor

CRT - Cathod Ray Tube

RGB - Reg Green Blue

BGR - Blue Green Red

vRGB - vertical Reg Green Blue

vBGR - vertical Blue Green Red

.NET - softwarová komponenta OS Windows

GDI - Graphics Device Interface

Zoznam príloh

A Dátový nosič CD s kompletnou implementáciou demonštračného programu a programového manuálu